Innovative Applications of O.R.

Mathematical analysis and solutions for multi-objective line-cell conversion problem

Yang Yu a, Jiafu Tang a, Jun Gong a, Yong Yin b, Ikou Kaku a,c,⇑

a Institute of Systems Engineering, State Key Laboratory of Synthetic Automation for Process Industries, Northeastern University, Shenyang, China
b Department of Economics and Business Management, Yamagata University, Yamagata, Japan
c Department of Environmental and Information Studies, Tokyo City University, Yokohama, Japan

A R T I C L E  I N F O

Article history:
Received 13 February 2012
Accepted 7 January 2014
Available online 20 January 2014

Keywords:
Assembly cell
Assembly line
Line-cell conversion
Multi-objective optimization
Genetic algorithm

A B S T R A C T

The line-cell (or line-seru) conversion is an innovation of assembly system applied widely in the electronics industry. Its essence is tearing out an assembly line and adopting a mini-assembly unit, called seru (or Japanese style assembly cell). In this paper, we develop a multi-objective optimization model to investigate two line-cell conversion performances: the total throughput time (TTPT) and the total labor hours (TLH). We analyze the bi-objective model to find out its mathematical characteristics such as solution space, combinatorial complexity and non-convex properties, and others. Owing to the difficulties of the model, a non-dominated sorting genetic algorithm that can solve large size problems in a reasonable time is developed. To verify the reliability of the algorithm, solutions are compared with those obtained from the enumeration method. We find that the proposed genetic algorithm is useful and can get reliable solutions in most cases.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

Today's business can be described by a single word: turbulence. Turbulent markets have the following characteristics: short product life cycles, uncertain product types, and fluctuating production volumes. To compete, sometimes only to survive, in such a volatile business environment, a number of approaches have been developed to aid companies for their management decisions and engineering designs (Yin, Kaku, Tang, & Zhu, 2011).

The line-cell (or line-seru) conversion, conceived at Sony, is an innovation of assembly system used widely in the Japanese electronics industry. To compete in a turbulent market, in 1992, several mini-assembly units were created in one of Sony's video-camera factories for an 8-millimeter CCD-TR55 video-camera, after dismantling a long assembly conveyor line. As did the original conveyor line, each mini-assembly unit produced the entire product. In 1994, Tatsuyoshi Kon, a former Sony staff, called this mini-assembly organization seru, a Japanese word for cellular organism. A detailed introduction of seru system and its managerial mechanism can be found in Yin, Stecke, and Kaku (2008) and Stecke, Yin, Kaku, and Murase (2012). Seru is similar to assembly cells, a widely adopted assembly system in western industries.

Equipment, however, is less important for seru. As a human-centered assembly system, seru is an old-fashioned workshop where crafts-person, including jack-of-all-trades workers, assembles an entire product from-start-to-finish without disruption. The assembly tasks are performed on fixed stations, so workers walk from station to station. A yatai is a single worker seru, the smallest production organization. So a yatai owner does all operational and managerial tasks by her- or himself (Liu, Lian, Yin, & Li, 2010; Liu, Stecke, Lian, & Yin, 2014; Stecke et al., 2012; Yu, Gong, Tang, Yin, & Kaku, 2012; Yu, Tang, Sun, Yin, & Kaku, 2013a, 2013b). In this paper, we only analyze rotating serus and yatais, and leave the analysis of divisional serus as a future research topic. To be consistent with previous research (Kaku, Gong, Tang, & Yin, 2008, 2009), from Section 2, we use “assembly cell” in this paper to represent seru (i.e., Japanese style assembly cell) and call the conversion from assembly lines to serus “line-cell conversion”.

⇑ Corresponding author at: Department of Environmental and Information Studies, Tokyo City University, Yokohama, Japan. Tel.: +81 45 910 2904; fax: +81 45 910 2605.
E-mail address: kakuikou@tcu.ac.jp (I. Kaku).

http://dx.doi.org/10.1016/j.ejor.2014.01.029
0377-2217/© 2014 Elsevier B.V. All rights reserved.
A seru system, which consists of one or more serus, is more flexible and agile than the assembly line. To improve the flexible of assembly lines, Guo, Wong, Leung, and Fan (2009) proposed an intelligent production control decision support system to solve the flexible assembly line (FAL) problem with flexible operation assignment. In addition, the seru system has a better balance than assembly line, because in seru system the balanced capacity can be improved by the workers assignment (i.e., cell formation) and batches assignment (i.e., cell loading) (Yu et al., 2012). To improve the balance of assembly line, Guo, Wong, Leung, Fan, and Chan (2008) used a genetic algorithm based optimization model to solve the assembly line balancing problem.

To respond to the uncertain demands from a turbulent market, a new management principle – Just-In-Time Organization System (JIT-OS) is used to manage a seru system. JIT-OS is different from the traditional, well-known Just-In-Time Material System (JIT-MS) of lean manufacturing. Yin et al. (2008) and Stecke et al. (2012) defined JIT-MS and JIT-OS as follows.

- "JIT-MS is a system that can provide the correct materials, in the right place, at the appropriate time, in the exact amount."
- "JIT-OS is a system that can provide the correct serus, in the right place, at the appropriate time, in the exact amount."

So JIT-OS is an extension or upgrade of JIT-MS from materials to organizations (i.e., serus). Most serus within a seru system can be constructed, modified, dismantled, and reconstructed frequently in a short time. This is a prerequisite for implementing a JIT-OS. For a detailed introduction of JIT-OS, see Stecke et al. (2012); for application cases of JIT-OS in Sony and Canon, see Yin et al. (2008) and Yin et al. (2012).

To fit the specific layout of a seru factory, appropriate case-by-case approaches are usually used to adjust the floor space of seru factories. Fortunately, since serus can be modified, dismantled, and constructed easily and quickly, and most serus occupy small spaces (e.g., mini units), managers can often get huge benefits from the adjustment of floor spaces. For example, by adopting seru system, Canon and Sony reduced 720,000 and 710,000 square meters of floor space, respectively (Stecke et al., 2012; Yin et al., 2012).

Cost can also be reduced largely by using seru systems. After adopting seru systems, Canon's costs were reduced significantly, by 55 billion yen in 2003, and by a total of 230 billion yen from 1998 to 2003. As a result, Canon emerged as a leading electronics maker. Its average productivity is higher than that of Toyota (Yin et al., 2008; Yin et al., 2012).

Other benefits from seru systems include (Stecke et al., 2012; Takeuchi, 2006) the reductions of throughput time, setup time, required labor hours, WIP inventories, and finished-product inventories. This paper analyzes two of seru performances: reductions in throughput time and required labor hours. Some amazing cases related to these two seru performances are, the throughput time was reduced by 53% at Sony Kohda and 35,976 required workers, equal to 25% of Canon's previous total workforce, have been saved (Yin et al., 2012). To investigate these two seru performances simultaneously, we adopt a bi-objective optimization model to study interactive relationships between them.

This paper, originally motivated by line-cell applications of Sony and Canon, has two purposes. First, we clarify several mathematical characteristics of the line-cell conversion problem. For simplicity and without loss of generality, we modify the model of Kaku et al. (2008) into a simple case in which a conveyor assembly line is converted to a pure seru (or cell) system. Second, we solve the developed multi-objective optimization problem. Two objectives of the total throughput time and the total labor hours are minimized simultaneously by using a NSGA-II based algorithm. We also use several numerical examples to illustrate the usefulness of our approach.

The remainder of this paper is organized as follows. The modified model is presented in Section 2. Section 3 clarifies several mathematical characteristics of the line-cell conversion problem from the modified model. Then a NSGA-II based algorithm is used to solve the multi-objective line-cell conversion problem in Section 4. Several numerical simulation experiments are performed to show the usefulness and performance of our approach in Section 5. Finally conclusions are given in Section 6.

2. A modified model of the line-cell conversion problem

2.1. Problem description

Kaku et al. (2008) compared three types of assembly systems: a pure cell system, a pure assembly line, and a hybrid assembly system that consists of an assembly line and several cells. For simplicity and without loss of generality, this paper studies a line-cell conversion problem shown in Fig. 1, i.e., an assembly conveyor line is converted to a pure cell system. All workers who formerly worked within the assembly line are assigned to cells (we name it as “assembly cell formation”).

A robust JIT-OS is the key for implementing a successful cell system. One important problem for designing a JIT-OS is to schedule or assign customer orders to different cells. We call this problem “assembly cell loading (ACL)”. Unfortunately, Yin, Stecke, Li, and Kaku (2011) have proved that even a simple ACL problem is NP-hard. In this paper, we adopt a First Come First Serve (FCFS) principle that applied in many companies. An arriving product batch is assigned to the empty cell with the smallest cell number. If all cells are occupied, the product batch is assigned to the cell with the earliest finish time. Fig. 2 shows a FCFS cell loading example with six batches and two cells. The length of rectangle charts in Fig. 2 is the flow time of a product batch.

We evaluate two line-cell conversion performances: throughput time and required labor hours, which have been reduced dramatically by seru users (e.g., 53% throughput time at Sony, 25% required workforce at Canon, respectively). Therefore, our problem is to decide how many cells should be formed, how to assign workers and product batches to appropriate cells to minimize two objectives, i.e., the total throughput time (TTPT) and the total labor hours (TLH).

2.2. Assumptions

Following assumptions are considered in this paper to construct the model of a pure cell system:

1. The types and batches of products are known in advance. There are $N$ product types that are divided into $M$ product batches. Each batch contains a single product type.
2. In the line-cell conversion process, the cost of duplicating equipment is ignored. Since most assembly tasks within a seru are manual so need only simple and cheap equipment. Duplicating them is usually not costly (Stecke et al., 2012; Yin et al., 2012).
3. A product batch needs to be assembled entirely within a single cell. In other words, a batch cannot be shared by cells.
4. Every assembly task is performed within its exclusive station. If a product type does not need some task, the product skips the task's station.
5. The assembly tasks within each cell are the same as the ones within the assembly line. In this paper, the number of tasks equals to $W$. 
6. A worker only performs a single assembly task in the assembly line (i.e., a specialist). In contrast, since the cells studied in this paper are rotating serus and yatais, a cell worker needs to perform all assembly tasks, assembles an entire product from-start-to-finish (i.e., a jack-of-all-trades), and there is no disruption or delay between adjacent tasks.

7. In the assembly line, each task (or station) is in the charge of a single worker.

8. The number of workers within each cell may be different, but not more than the total number of workers.

9. Setup time is considered when two different product types are assembled consecutively; otherwise the setup time is zero.

2.3. Notations

We define the following terms:

- **Indices**
  - $i$: Index of workers ($i = 1, 2, \ldots, W$).
  - $j$: Index of cells ($j = 1, 2, \ldots, J$).
  - $n$: Index of product types ($n = 1, 2, \ldots, N$).
  - $m$: Index of product batches ($m = 1, 2, \ldots, M$).
  - $k$: Index of the sequence of product batches in a cell ($k = 1, 2, \ldots, M$).

- **Parameters**
  - $V_{mn} = \begin{cases} 1, & \text{if product type of product batch } m \text{ is } n \\ 0, & \text{otherwise} \end{cases}$
  - $B_m$: Size of product batch $m$.
  - $T_n$: Cycle time of product type $n$ in the assembly line.
  - $SL_n$: Setup time of product type $n$ in the assembly line.
  - $SCP_n$: Setup time of product type $n$ in a cell.
  - $g_i$: Upper bound on the number of tasks for worker $i$ in a cell. If the number of tasks assigned to worker $i$ is more than $g_i$, worker $i$’s average task time within a cell will be longer than her or his task time within the original assembly line.
  - $CW_i$: Coefficient of variation of worker $i$’s increased task time after line-cell conversion, i.e., from a specialist to a completely cross-trained worker.
  - $e_i$: Worker $i$’s coefficient of influencing level of doing multiple assembly tasks.
  - $b_{ni}$: Skill level of worker $i$ for each task of product type $n$.

- **Decision variables**
  - $X_{ij} = \begin{cases} 1, & \text{if worker } i \text{ is assigned to cell } j \\ 0, & \text{otherwise} \end{cases}$
  - $Z_{mjk} = \begin{cases} 1, & \text{if product batch } m \text{ is assigned to cell } j \text{ in sequence } k \\ 0, & \text{otherwise} \end{cases}$

In addition, if $k = 0$, $Z_{mjk} = 0$.

- **Variables**
  - $SC_m$: Setup time of product batch $m$ in a cell.
  - $TC_m$: Assembly task time of product batch $m$ per station in a cell.
  - $FC_m$: Flow time of product batch $m$ in a cell.
  - $FCB_m$: Begin time of product batch $m$ in a cell.

2.4. Problem formulation

We consider an assembly planning problem in which there is an assembly product mix with $M$ product batches and $N$ product types. $W$ workers are assigned to assembly cells during the
line-cell conversion. The batches are assigned to cells with the FCFS principle. We define the total throughput time of the cell system following this FCFS principle.

First, the cross-training process can be represented as a V-shaped learning curve. In other words, in the early period of the line-cell conversion, workers often cost more time on tasks she or he is not familiar with (Yin et al., 2012). So it is reasonable to assume that a worker’s skill level varies with the number of tasks assigned to her or him. In this paper, we assume that if the number of worker’s tasks within a cell is larger than her or his upper bound η, i.e., W > η, then the worker will spend more average task time than her or his task time within the original assembly line. The details are given as Eq. (1).

\[ CW_i = \begin{cases} 1 + e_i(W - η_i), & W > η_i, \\ 1, & W \leq η_i \end{cases}, \quad ∀i \]

Second, the task time of a product varies with workers’ skill levels. Therefore, for a cell, the task time of a product is calculated by average task time of workers in the cell. The task time of product batch \( m \) per station in a cell can be represented by the following equation.

\[ TC_m = \frac{\sum_{i=1}^{N} \sum_{j=1}^{W} X_{ij} V_{m_i} T_{βm_i} CW_{X_{mjk}}}{\sum_{i=1}^{W} \sum_{j=1}^{m} X_{ij} Z_{mjk}} \]

Finally, the setup time \( SC_m \), the flow time \( FC_m \) and the begin time \( FCB_m \) of product batch \( m \) are represented as below.

\[ SC_m = \sum_{m=1}^{N} SC_m V_{m_i} \left( 1 - \sum_{m=1}^{N} X_{m_i} Z_{mjk} \right), \quad ∀m \]

\[ FC_m = \frac{B_{m} TC_{m} W}{\sum_{i=1}^{W} \sum_{j=1}^{m} X_{ij} Z_{mjk}} \]

\[ FCB_m = \sum_{m=1}^{M} \sum_{i=1}^{W} X_{ij} FC_{m} + SC_{m} Z_{mjk} \]

Eq. (3) states the setup time of product batch \( m \). Setup time is considered when two different types of products are processed consecutively; otherwise the setup time is zero. This is a real-life consideration. One of the authors visited three companies’ (Omron, Yamaha, and Fujitsu) assembly cell factories recently and observed the above case. Eq. (4) states the flow time of product batch \( m \) within a cell. Eq. (5) states the begin time of each product batch. There is no waiting time between two product batches so that the begin time of one product batch is the aggregation of flow time and setup time of all of the previous product batches that are in the same cell.

The multi-objective mathematical model is given in Eqs. (6)–(12) as below.

**Objective functions**:

\[ \text{min} \{ \text{max}(FCB_m + FC_m + SC_m) \} \]  

\[ \text{min} \sum_{m=1}^{W} \sum_{i=1}^{m} \sum_{j=1}^{M} X_{ij} Z_{mjk} \]

**Subject to**:

\[ \sum_{i=1}^{W} X_{ij} \leq W, \quad ∀j \]

\[ \sum_{j=1}^{M} X_{ij} = 1, \quad ∀i \]

\[ \sum_{i=1}^{W} \sum_{j=1}^{M} X_{mjk} = 1, \quad ∀m \]

\[ \sum_{m=1}^{M} \sum_{i=1}^{W} X_{ij} = 0, \quad ∀j \]

\[ \sum_{i=1}^{W} \sum_{j=1}^{M} Z_{mjk} = \sum_{i=1}^{W} \sum_{j=1}^{M} Z_{mjk} \]

where Eq. (6) states the objective to minimize the total throughput time (TPTT) of all product batches. The TPTT is the finish time of the last completed product batch. Eq. (7) states the objective to minimize the total labor hours (TLH) of all product batches. The TLH is the cumulative working time of all workers in the cell system. Eq. (8) is the number constraint because the number of workers within a cell cannot exceed the total number of available workers (\( W \)). Eq. (9) is the worker assignment rule, i.e., each worker should be assigned to one and only one cell. Eq. (10) is the product batch assignment rule, i.e., each batch should be assigned to one and only one cell. Eq. (11) ensures that a product batch cannot be assigned to an empty cell. Eq. (12) means that product batches must be assigned sequentially.

### 3. Model analysis

It is easy to see that the line-cell conversion model created in Section 2.4 is not linear. To find hints for solving the developed model, we need to clarify the characteristics of the line-cell conversion problem in Section 3.1. We analyze the properties of the line-cell conversion problem’s solution space and combinatorial complexity in Section 3.2. The property of Pareto-optimal front in our developed model is discussed in Section 3.3.

#### 3.1. Model characteristics

**Property 1.** Regardless of the cell loading’s scheduling rule, the worker assignment in a cell system is an assignment problem.

**Explanation.** Consider \( C \{i, j, k \} \) denotes cell \( j \) which assigned \( n \) workers to, and then \( X_{ij} = 1 \) if \( i \in C \) of order of \( n \) workers. For example, consider a cell \( j \) with 2 workers labeled as 1 and 2, and both \( X_{ij} \) and \( X_{ik} \) are always 1, regardless of the order of the two workers (i.e., \( X_{ij} \) is independent of the order of \( n \) workers). So according to Eqs. (3)–(5), if other conditions remain unchanged, \( SC_m \), \( FC_m \) and \( FCB_m \) are not influenced by the order of workers assigned to each cell. Similarly, from Eqs. (6) and (7), TPTT and TLH are independent of the order of workers assigned to each cell. Given the order of cells and batches, the changing order of workers in any cell does not influence the objectives of line-cell conversion.

**Property 2.** For a line with \( W \) workers, the feasible formations of a cell can be expressed by the nonempty subsets of the set \( \{1, 2, \ldots, W\} \).

**Explanation.** Let \( X = \{1, 2, \ldots, W\} \) stand for the set of all workers. Let \( C \) stand for an arbitrary feasible formation of a cell. \( C \) is non-empty, i.e., \( C \) contains at least one worker (see Eq. (8)). Therefore, \( C \) can be expressed as \( C = \{i | i \in X\} \), i.e., \( C \subseteq X \). In addition, \( C \) is independent of the order of workers assigned to the cell (see Property 1), so \( C \) is one of nonempty subsets of the set \( X \). Since \( C \) is an arbitrary feasible formation of a cell, this means that feasible formations of a cell can be express by the nonempty subsets of the set \( \{1, 2, \ldots, W\} \).
For example, consider a line with 2 workers, the all 3 feasible formations of a cell are following: (1) (this means worker 1 is assigned to a cell), (2) and (1,2). Obviously, they are the nonempty subsets of the set \{1,2\}.

**Theorem 1.** The line-cell conversion problem is NP-hard.

**Proof.** The line-cell conversion includes the assembly cell formation (ACF) and the assembly cell loading (ACL) problem. The ACF is to partition \(W\) workers of an assembly line into pairwise disjoint nonempty cells. We show that the ACF is an exact cover problem, which is NP-complete and is one of Karp’s 21 NP-complete problems (Karp, 1972). □

In mathematics, given collection \(S\) of nonempty subsets of a set \(X\), an exact cover of \(X\) is a subcollection \(S'\) of \(S\) that satisfies the following two conditions: (1) the sets in \(S'\) are pairwise disjoint; and (2) the union of the sets in \(S'\) covers \(X\).

Let \(X\) stand for the set of all workers, so the cardinality of \(X\), \(|X| = \frac{W}{|W|}\) (i.e., the number of workers). Let \(P\) stand for an arbitrary solution of the ACF problem, so \(P\) is a set whose elements are nonempty cells. Since nonempty cells are expressed as the nonempty subsets of the set \(X\) (proved by Property 2), \(P\) can be expressed as \(P = \{x| x \subseteq X\}\). The cardinality of \(P\), \(|P| = 1, 2, \ldots, W\) (i.e., the number of cells). We have two cases.

Case 1: \(|P| = 1\). This case means that all workers are assigned to the same cell, i.e., \(P = X\). In mathematical words, set \(X\) is an exact cover of itself.

Case 2: \(2 \leq |P| \leq W\).

Suppose \(A \in P, \emptyset \not= B \in P \text{ and } A \neq B\). Then \(A \cap B = \emptyset\) (because cells are pairwise disjoint). Let \(y\) stand for an arbitrary worker, i.e., \(y \in X\). Since all workers are assigned to cells, we can find a cell \(C \in P\) such that \(y \in C\). Since \(y\) is an arbitrary worker, obviously we can get \(\cup P = \{y|\exists C \in P \text{ and } y \in C\} = X\), which means that \(\cup P\) covers all of \(X\).

Both \(A\) and \(B\) are arbitrary cells of \(P\); they are non-overlapping and nonempty, and \(\cup P\) covers all of \(X\), we can conclude that \(P\) is an exact cover of \(X\).

Take cases 1 and 2 together, we can conclude that \(P\) is an exact cover of \(X\). Since \(P\) is an arbitrary solution of the ACF, this means that the ACF is an exact cover problem, which is NP-complete (Karp, 1972).

Similarly, Yin, Kaku et al. (2011) have proven that even a simple version with the FCFS rule, the solutions of line-cell conversion are independent of the sequence numbers of the formatted cells. Therefore \(SC_m, FC_m, FCB_m, TTPT\) and TLH are independent of the sequence of the formatted cells. □

3.2. The solution space and combinatorial complexity

**Theorem 2.** Using the FCFS rule in the cell loading, the assignment of sequence numbers to cells is a permutation problem.

**Proof.** Consider \(\{C_1, C_2, \ldots, C_j\}\) be a sequence of \(J\) formatted cells. If there are empty cells which are not assigned any batch to, an arriving product batch is assigned to the empty cell with the smallest cell number. When all cells are occupied, a product batch is assigned to the cell with the earliest finish time as shown in Fig. 2. Given the sequence of batches and the FCFS rule, the changing sequence numbers of the \(J\) cells will produce the different \(Z_{mk}\) that influences the values of \(SC_m, FC_m\) and \(FCB_m\) (see Eqs. (3)–(5)). So TTPT and TLH are dependent on the sequence numbers of the formatted cells (see Eqs. (6) and (7)). Therefore, given the sequence of batches, the changing sequence numbers of the formatted cells does influence the objectives of line-cell conversion. □

For example, consider an assembly line with 3 workers labeled as 1, 2 and 3. \([[1,2],[3]]\) is a solution of the line-cell conversion problem, which is not equivalent to \([[3],[1,2]]\) by the FCFS rule.

**Theorem 3.** Using the SPT rule in the cell loading, the assignment of sequence numbers to cells is a combination problem.

**Proof.** Consider \(\{C_1, C_2, \ldots, C_j\}\) be a sequence of \(J\) formatted cells. Since a batch will be always assigned to the cell with the shortest process time for it by the SPT rule, given the sequence of batches and the SPT rule, the \(Z_{mk}\) is independent of the sequence of the formatted cells. Therefore \(SC_m, FC_m, FCB_m, TTPT\) and TLH are independent of the sequence of the formatted cells. □

For example, consider an assembly line with 3 workers labeled as 1, 2 and 3. \([[1,2],[3]]\) is a solution of the line-cell conversion problem, which is equivalent to \([[3],[1,2]]\) by the SPT rule.

**Theorem 4.** The solution space of the line-cell conversion with \(W\) workers with the FCFS rule is the same as the ordered partition of set \(W\) elements.

**Proof.** In mathematics, a partition of a set \(X\) is a division of \(X\) into pairwise disjoint nonempty subsets that cover all of \(X\). More formally, these subsets are both collectively exhaustive and mutually exclusive with respect to the set being partitioned (Brualdi, 2004). Theorem 1 shows an arbitrary solution \((P)\) of ACF is a partition of a set \(X\) with \(W\) workers, because cells in \(P\) are pairwise disjoint and \(\cup P = X\). □

In the ordered set partition, the solutions are independent of the order of the elements in the subsets, but dependent on the order of the subsets. Given the number of workers \(W\) and with the FCFS rule, the solutions of line-cell conversion are independent of the order of the workers within the cells (i.e., assignment problem proved by Property 1), but dependent on the order of the cells (i.e., permutation problem proved by Theorem 2).

We show the detailed result of the solutions of the line-cell conversion with the FCFS rule. Consider an line with 3 workers labeled as 1, 2 and 3, the all 13 solutions of the line-cell conversion with the FCFS rule are following: \([[1,2],[3]], [[1,2],[3]]\) (this means 2 cells are constructed in which workers 1 and 2 are in cell 1, worker 3 is in cell 2), \([[3],[1,2]]\), \([[1,3],[2]], [[2,1],[3]]\), \([[2,3],[1]]\), \([[1,2],[3]]\), \([[1,2],[3]]\), \([[1,3],[2]], [[1,2],[3]]\), \([[2],[1],[3]]\), \([[2],[1],[3]]\), \([[3],[1],[2]]\) and \([[3],[2],[1]]\). Obviously, this is the same as an ordered set partition of the set \(\{1,2,3\}\).

The complexity of the line-cell conversion with the FCFS rule can be expressed by the number of ordered set partition, sometimes called ordered Bell numbers (Carlitz, 1964). It can be represented as follows.

\[
F(W) = \sum_{k=1}^{W} F(W, k) k!
\]

where \(F(W, k)\) is the number of the solutions of converting a line with \(W\) workers to \(k\) cells with the FCFS rule and equal to \(S(n,k)\) in Stirling numbers of the second kind (Rennie & Dobson, 1969). \(F(W)\) is the
number of the solutions of converting a line with W workers to cells with the FCFS rule. The numbers of \( F_1 \sim F_{10} \) are 1, 3, 13, 75, 541, 4683, 47,293, 545,835, 7,087,261 and 102,247,563, respectively.

**Theorem 5.** The solutions space of the line-cell conversion with W workers with the SPT rule is the same as the unordered partition of set with W elements.

**Proof.** In the unordered set partition, the solutions are independent of the order of the elements in the subsets and the order of the subsets. Given the number of workers (W) and with the SPT rule, the solutions of line-cell conversion are independent of the order of the workers within the cells (i.e., assignment problem proved by Property 1) as well as the order of the cells (i.e., combination problem proved by Theorem 3). \( \square \)

We show the detailed result of the solutions of the line-cell conversion with the SPT rule. Consider an line with 3 workers labeled as 1, 2 and 3, the all 5 solutions of the line-cell conversion with the SPT rule are following: \([1,2,3],[1,2],[3],[2,3],[1]\) (this means 2 cells are constructed in which workers 1 and 2 are in cell 1, worker 3 is in cell 2), \([1,3],[2],[3,1]\) and \([1,2],[2,3],[3]\). Obviously, this is the same as unordered set partition of the set \([1,2,3]\).

The complexity of the line-cell conversion with the SPT rule can be expressed by the number of unordered set partition, sometimes called unordered Bell numbers (Klazar, 2003; Knopfmacher & Mays, 2006; Williamson, 1985). It can be found recursively as follows.

\[ S(W + 1) = \sum_{k=0}^{W} \binom{W}{k} S(k) \]

where \( S(W) \) is the number of the feasible solutions of converting a line with W workers to cells with the SPT rule. The numbers of \( S_1 \sim S_{10} \) are 1, 2, 5, 15, 52, 203, 877, 4140, 21,147 and 115,975, respectively.

**Theorem 1** means that the model described in this paper is in general computationally intractable.

Eqs. (13) and (14) state that the number of feasible solutions increases exponentially with the number of workers. For large size problems, however, there is no efficient algorithm for solving the problem to optimality, unless \( P = NP \). There are two ways to overcome such computational difficulties. The first way is to analyze specific line-cell conversion models, find optimal solutions by uniquely developed approaches (e.g., modified branch and bound, cutting approaches and others). It is a future work of our research. The second way is using evolutionary algorithms to find a better solution within a reasonable computational time, the purpose of this study. In this paper, we only consider the line-cell conversion problem with the FCFS rule. A comparative study of scheduling rules (e.g., FCFS vs. SPT) for the line-cell conversion problem is discussed in the paper (Yu, Tang, Li, Yin, & Kaku, 2011).

### 3.3. Pareto-optimal front

It is desirable to obtain an optimal solution that not only minimizes the TTPT but also the TLH. Unfortunately, it is not easy. Generally an approach is good enough if it can provide the Pareto-optimal or non-dominated solutions, which can be used to determine the appropriate conversion plans. For instance, consider an assembly line with 5 workers labeled as 1, 2, 3, 4 and 5. Two solutions for this problem are \( S_1 = \{[3],[2],[1,5],[4]\} \) (i.e., four cells are constructed in which worker 3 is in cell 1, worker 2 is in cell 2, and so one) and \( S_2 = \{[2],[3],[1,5],[4]\} \). According to the data described in Section 5.1 and Eq. (6), the TTPT for the two solutions are the same: 2981.485. However, from Eq. (7), the TLH of \( S_1 \) and \( S_2 \) are 14509.845 and 14554.665, respectively. That is, \( S_1 \) dominates \( S_2 \).

Similarly, consider other two solutions, \( S_3 = \{[1,5],[2],[3],[4]\} \) and \( S_4 = \{[1,5],[3],[2],[4]\} \), the TTPT for these two solutions are the same and equal to 2986.188. However, the TLH of \( S_3 \) and \( S_4 \) are 14509.845 and 14554.665, respectively. That is, \( S_3 \) dominates \( S_4 \). Note that the TTPT of \( S_1 \) is smaller than the one of \( S_3 \), whereas the TLH of \( S_1 \) is bigger than the one of \( S_3 \), the \( S_1 \) and \( S_3 \) pair are called non-dominated solutions. The non-dominated solutions of all feasible solutions form the so-called non-dominated or Pareto-optimal front.

The non-dominated or Pareto-optimal front provides an opportunity to find a conversion plan that minimizes the total throughput time (TTPT) and the total labor hours (TLH). Fig. 3 illustrates the non-dominated front found by an enumeration algorithm for the above example.

For the assembly line of the above example, its TTPT and TLH are 3312.634 and 16255.17 respectively. In Fig. 3, for the non-dominated solution with the minimum TTPT, TTPT is 2929.79 and 11.56% = \([329.79 - 329.79] / 329.79 \times 100\%\) better than line, and TLH is 14512.23 and 10.71% better than line. For the non-dominated solution with the minimum TLH, TLH is 14456.43 and 11.67% better than line, and TTPT is 3270.56 and 1.27% better than line. So the line-cell conversion can be used to improve simultaneously the assembly system’s performance of the TTPT and TLH.

**Theorem 6.** The non-dominated front of the line-cell conversion with the FCFS rule is non-convex.

**Proof.** From the graphs of non-dominated solutions in Fig. 3, it is easy to prove that the Pareto-optimal front of the multi-objective line-cell conversion is non-convex. Take the two points of the non-dominated front as \( x_1 \) and \( x_2 \) (for example, the points #1 and #6 in Fig. 3 numbered from left side), then there exists a parameter \( 0 < \lambda < 1 \) such that \( \lambda x_1 + (1 - \lambda) x_2 \) is not belong to the non-dominated front, it proves the theorem. \( \square \)

The conventional approach for solving a multi-objective problem is to convert it into a single objective problem by using a weighted sum of objectives. However, this approach has a drawback. Certain optimal solutions may be lost since they may never be explored, particularly when the non-convex objective function gives rise to a duality gap (Goicoechea, Hansen, & Duckstein, 1982). Hence, such approach cannot be applied to find Pareto-optimal solutions for a line-cell conversion problem that has a non-convex non-dominated or Pareto-optimal front. On the contrary, the multi-objective evolutionary algorithms (MOEAs) are widely used because they can find widespread Pareto-optimal solutions in a single simulation run (Deb, 2001).

#### 3.4. Multi-objective evolutionary algorithms

A number of multi-objective evolutionary optimization algorithms have been developed since the pioneering work of Schaffer

![Fig. 3. The non-dominated (Pareto-optimal) front for the example with 5 workers.](image-url)
(1984), Schaffer (1985). Instead of a single solution, evolutionary optimization algorithms (Coutinho-Rodrigues, Tralhão, & Alcada-Almeida, 2012; Cruz-Ramirez, Hervás-Martinez, Fernández, Briceño, & Mata, 2012) produce a population of solutions. Since multi-objective optimization problems give rise to a set of Pareto-optimal solutions, evolutionary optimization algorithms are ideal for handling multi-objective optimization problems. Recently, several evolutionary approaches have been proposed to deal with the multi-objective optimization problems. They are non-dominated sorting genetic algorithm (NSGA-II) (Deb, Pratap, Agarwal, & Meyarivan, 2002; Gao et al., 2008), strength Pareto evolutionary algorithm (SPEA) (Zitzler, 1999), Pareto archived evolution strategy (PAES) (Knowles & Corne, 1999), multi-objective differential evolution (MODE) (Ali, Siarry, & Pant, 2012; Xue, Sanderson, & Graves, 2003), and others. In addition, the hybrid optimization algorithms can be used to effectively solve the multi-objective problems, e.g., the multi-objective flexible job-shop scheduling problem (Xia & Wu, 2005) and the balancing control problem of the hybrid assembly lines (Zeng, Wong, & Leung, 2012).

The well-known non-dominated sorting genetic algorithm II (NSGA-II) (Deb et al., 2002; Lin & Yeh, 2012) is widely used. Compare with other multi-objective genetic algorithms (Coello, de Computacion, & Zacatenco, 2006), NSGA-II provides excellent results. Here we apply the NSGA-II approach to find the optimal solution for multi-objective line-cell conversion problems. A brief description of NSGA-II is represented as follows (Wang, Hirschb, Kanga, & Lacorc, 2011).

Initially, a parent population $P_0$ of size $n$ is generated randomly by the general genetic algorithm, where $n$ is the population size. All individuals in this population are sorted into different front levels based on the domination of pair comparison. Each front level is assigned a fitness (or a rank) that equals its non-dominating level. The first front is the completely non-dominant set in the current population and the second front is dominated by the individuals in the first front only, and so on. Each individual in a front is assigned a rank (fitness) value. Individuals in the first front are given a fitness value of 1 and individuals in the second front are assigned fitness value as 2, and so on. In addition, a parameter called crowding distance is calculated for each individual. The crowding distance is a measure of how close an individual is to its neighbors. A long average crowding distance means a better diversity in the population. The definition of the crowding distance is as follows (Deb et al., 2002; Wang et al., 2011):

$$C_j = \sum_{i=1}^{N_{\text{obj}}} \left( \frac{F_{i,j+1}^{-1} - F_{i,j}^{-1}}{(F_{i}\text{max})_{j} - (F_{i}\text{min})_{j}} \right)$$

where $N_{\text{obj}}$ is the number of objectives, $F_{i,j}$ is the value of the $i$th objective function of the $j$th individual in a non-dominated front. If there are two objectives of $F_1$ and $F_2$, the crowding distance of the $j$th solution is the averaged cuboids’ length as illustrated in Fig. 4 (Deb et al., 2002; Wang et al., 2011). For boundary points, the crowding distance is set as the maximum value to ensure these points can survive to the next generation.

After sorting of population $P_0$, parents are selected from the population by using binary tournament selection based on the rank and crowding distance. The offspring population $Q_0$ of the same size $n$ is generated by crossover and mutation operators, which will be discussed in detail in a later section. Thereafter, $P_0$ and $Q_0$ are combined into a whole population $C_0$ of size $2n$. The population $C_0$ is sorted again based on non-dominination and only the best $n$ individuals are selected to fill the new parent population $P_1$. The selection is also based on front level and the crowding distance on the last front. Then the procedure goes on, iterating till $N$, the maximum number of generations or unchanging non-dominated front, is reached. The original NSGA-II can be described as the following steps (Wang et al., 2011):

1. Initialize $P_0$ of size $n$ and sort $P_0$ based on non-dominance.
2. Generate the $Q_0$ of size $n$ by a classical GA.
3. Combine $P_0$ and $Q_0$ into $C_0$ of size $2n$ and sort $C_0$ based on non-dominance.
4. Fill $P_1$ with non-dominated individual in $C_0$.
5. Repeat Step 2 to Step 4 until $N$ is reached.
6. Output non-dominated front.

Gao et al. (2008) show that NSGA-II is able to maintain a better spread of solutions and converge better than other multi-objective evolutionary algorithms. It has been successfully applied to many multi-objective optimization problems. Hence this paper presents an algorithm based on NSGA-II to solve the multi-objective line-cell conversion optimization problem.

4. A NSGA-II based algorithm

In this study, we modify the original NSGA-II to solve the multi-objective line-cell conversion optimization problem. The binary tournament selection operator proposed by Deb and Agrawal (1995) is adopted. However, the chromosome coding, crossover operation mutation operation, and the combination of parent population and offspring population are modified to fit our purpose.

4.1. Chromosome coding

The chromosome with $n$ workers is represented by a vector with $2n - 1$ elements. An element presents a worker number if its value is not more than $n$; otherwise, it presents the separating character. We assume that there are two fixed separating characters in the start and end locations of the vector. If there exists at least one worker between two separating characters, a cell that contains the worker(s) is constructed. To illustrate how this works, consider the following two sequences with 5 workers:

Chromosome 1: 1 1 7 5 8 3 9 2 6 4
Chromosome 2: 8 1 7 5 3 9 6 2 4

In the first chromosome, 7, 8, 9 and 6 are the separating characters, so 5 cells are constructed where worker 1 is in cell 1, worker 5 is in cell 2, and so on.

In the second chromosome, 8, 7, 9 and 6 are the separating characters, so 3 cells are set up where worker 1 is in cell 1, workers 5 and 3 are in cell 2, and workers 2 and 4 are in cell 3.

4.2. Evaluation of solutions

A solution $S_i$ for a bi-objective line-cell conversion is computed by the values of the objectives TTPT based on Eq. (6) and TLH based
on Eq. (7). The fitness of the solution \( S_i \) is evaluated by the non-domination rank and crowding distance of NSGA-II.

### 4.3. Selection

The sorting is based on the non-domination rank and crowding distance. The individuals are selected by using a binary tournament selection. Two solutions are picked from the population. After comparison, the better solution is chosen. An individual is selected if its rank is lesser than others; or if its crowding distance is greater than the others with the same rank (Beyer & Deb, 2001).

### 4.4. Crossover

There are several crossover techniques available for general sequencing problems, e.g., partially-mapped crossover (PMX; Goldberg & Lingle, 1985), cycle crossover (CX; Oliver, Smith, & Holland, 1987) and immediate successor relation crossover (ISRX; Hyun, Kim, & Kim, 1998). Here we use the Order Crossover (OX)

- The two underlined elements are selected randomly for swap- ping. After mutation, the sequence is as follows:
  - Before mutation: 1 2 5 8 3 9 2 6 4
  - After mutation: 1 2 5 8 3 9 7 6 4

### 4.5. Mutation

For protecting diversity of a population, mutation is sometimes performed so that a solution has an occasional trait that is unique from its parents. In this paper, mutation is the swapping of two unique elements in a sequence. Consider the following sequence:

- The two underlined elements are selected randomly for swapping. After mutation, the sequence is as follows:
  - After mutation: 1 2 5 8 3 9 7 6 4

### 4.6. The combination of parent and offspring populations

The aim of the improved combination of parent and offspring populations is to figure out the exact Pareto points effectively and to maintain diversity in the meantime. The combination of parent and offspring populations in original NSGA-II algorithm is described in Section 3.3. \( P_i \) and \( Q_i \) are combined into a whole population \( C_i \) of size 2n. Because \( Q_i \) is generated from \( P_i \), they have many same solutions. \( C_i \), which is combined from \( P_i \) and \( Q_i \), has many same solutions. To protect diversity of the population, only those individuals of \( Q_i \) that are different from \( P_i \) are combined into the new population \( C_i \). The proposed combination algorithm of parent and offspring populations is presented as follows.

#### Input: parent population \( P_i \) and offspring population \( Q_i \).

#### Output: new population \( C_i \).

1. Initialize \( C_i = P_i \).
2. For each individual \( q \) in \( Q_i \) do the following:
   - If \( q \) dominates \( P_i \), go to step 3.
   - If \( q \) does not dominate \( P_i \), go to step 3.
   - (2-1) Initialize \( F_q = \) false. This bool would denote if the individual is in \( P_i \).
   - (2-2) For each individual \( p \) in \( P_i \):
     - (2-2-1) If \( q.TPT = p.TPT \) and \( q.TLH = p.TLH \) then \( F_q = \) true; Break.
   - (2-3) if \( F_q = \) false then add \( q \) into \( C_i \).

From above description, we can see that \( C_i \) always includes \( P_i \) and the solutions of \( Q_i \) that are different from \( P_i \). So the larger crossover probability \( (P_c) \) or mutation probability \( (P_m) \) may produce more solutions of \( Q_i \) that are not in \( P_i \), which accelerates the speed of population towards the true non-dominated front.

### 5. Computational experiments

#### 5.1. Test instances

To test the developed algorithm, we use simulation examples that are generated by simulating line-cell conversion applications we investigated (e.g., Canon, Fujitsu). The experimental data are described in Tables 1–5. They show the parameters of the experiment we investigated (e.g., Canon, Fujitsu). The experimental data are described in Tables 1–5. They show the parameters of the experiment we investigated (e.g., Canon, Fujitsu).

#### 5.2. Parameter setting

According to the description in Section 4.6, increasing \( P_c \) and \( P_m \) may produce a better non-dominated front. In this paper, the crossover probability \( (P_c) \) and the mutation probability \( (P_m) \) are fixed to 0.5 and 0.9, respectively.
5.3. Hardware and software specifications

The developed NSGA-II based algorithm and the Enumeration were coded in C# and executed on an Intel Core(TM) 2 processor of 3.0 Gigahertz under Windows XP using 992 Megabytes of RAM.

5.4. Results of the small instances

There are 541 solutions obtained by Enumeration for the instance with 5 workers shown in Fig. 5, where non-dominated solutions are seven. The solution of \(\{3,4,5\}\) has a minimal TLH, and the solution of \(\{(1,2),(3,4,5)\}\) has a minimal TPTT. However, because non-dominated solutions are more than 1, we cannot say which is better. Fig. 6 shows 4683 solutions obtained by Enumeration for the instance with 6 workers, where non-dominated solutions are 9.

Fig. 7 shows the initial and final populations of the NSGA-II based algorithm for the instance with 6 workers. It can be observed that the solutions are converged into a set. And the obtained non-dominated front is the same as the result from enumeration.

5.5. Comparative results with enumeration

We give a detail comparative discussion here. The computational complexity of non-dominated sorting is \(O(MN^2)\), where \(M\) is the number of objectives and \(N\) is the number of solutions. If the number of workers is 7, by Eq. (13), all of feasible solutions with the FCFS rule are 47,293 and the complexity of non-dominated sorting for the solutions is 4,473,255,698. Hence when the number of workers is larger than 6, using Enumeration is difficult to get the true non-dominated front. Therefore, to test the performance of our proposed NSGA-II based algorithm, we solve examples of 5 and 6 workers by our algorithm and compare the results to the true non-dominated fronts obtained from enumeration.

Hansen and Jaszkiewicz (1998) discussed several measures and proposed a general framework to evaluate an approximation set of the true non-dominated or Pareto-optimal set (the reference set \(R\)). The ratio of non-dominated individuals (\(RNI\)) (Tan, Lee, & Khor, 2001) is defined as the proportion of the number of efficient solutions in the approximate set to the number of solutions in \(R\). Note that this proportion can be zero and yet \(A\) can be a good approximation. In addition, distances measure the average/maximum distance between the points \(A\) and \(R\) are as follows (Czyak & Jaszkiewicz, 1998; Ulungu, Teghem, & Ost, 1998):

\[
D_{\min} = \frac{1}{|R|} \sum_{z \in R} \min_{z' \in ND} (z', z) \tag{16}
\]

\[
D_{\max} = \max_{z \in R} \left\{ \min_{z' \in ND} (z', z) \right\} \tag{17}
\]

where \(ND\) is the non-dominated set generated by the algorithm and \(d\) is defined by

Table 1

<table>
<thead>
<tr>
<th>Factor</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product types</td>
<td>5</td>
</tr>
<tr>
<td>Batch size</td>
<td>(N(50,5))</td>
</tr>
<tr>
<td>(\epsilon_i)</td>
<td>(N(0.2,0.05))</td>
</tr>
<tr>
<td>SL</td>
<td>2.2</td>
</tr>
<tr>
<td>SCP</td>
<td>1.0</td>
</tr>
<tr>
<td>(T_n)</td>
<td>1.8</td>
</tr>
<tr>
<td>(\eta_i)</td>
<td>10</td>
</tr>
</tbody>
</table>

\(N(50,5)\): Normal distribution \((\mu = 50, \sigma = 5)\).

Table 2

<table>
<thead>
<tr>
<th>Worker</th>
<th>(\epsilon_i)</th>
<th>(\epsilon_i)</th>
<th>(\epsilon_i)</th>
<th>(\epsilon_i)</th>
<th>(\epsilon_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.18</td>
<td>0.19</td>
<td>0.21</td>
<td>0.22</td>
<td>0.19</td>
</tr>
<tr>
<td>2</td>
<td>0.18</td>
<td>0.23</td>
<td>0.24</td>
<td>0.22</td>
<td>0.18</td>
</tr>
<tr>
<td>3</td>
<td>0.18</td>
<td>0.23</td>
<td>0.24</td>
<td>0.22</td>
<td>0.18</td>
</tr>
<tr>
<td>4</td>
<td>0.18</td>
<td>0.23</td>
<td>0.24</td>
<td>0.22</td>
<td>0.18</td>
</tr>
<tr>
<td>5</td>
<td>0.18</td>
<td>0.23</td>
<td>0.24</td>
<td>0.22</td>
<td>0.18</td>
</tr>
</tbody>
</table>

Table 3

Table 4

Table 5

Table 2

Worker \(i\)’s coefficient of influencing level of doing multiple assembly tasks \((\epsilon_i)\).

<table>
<thead>
<tr>
<th>Worker</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.92</td>
<td>0.96</td>
<td>1.04</td>
<td>1.09</td>
<td>1.2</td>
</tr>
<tr>
<td>2</td>
<td>0.95</td>
<td>0.97</td>
<td>1.09</td>
<td>1.12</td>
<td>1.18</td>
</tr>
<tr>
<td>3</td>
<td>0.99</td>
<td>1.01</td>
<td>1.05</td>
<td>1.09</td>
<td>1.21</td>
</tr>
<tr>
<td>4</td>
<td>1.03</td>
<td>1.07</td>
<td>1.09</td>
<td>1.12</td>
<td>1.25</td>
</tr>
<tr>
<td>5</td>
<td>0.96</td>
<td>1.02</td>
<td>1.05</td>
<td>1.1</td>
<td>1.18</td>
</tr>
<tr>
<td>6</td>
<td>1.01</td>
<td>1.1</td>
<td>1.1</td>
<td>1.15</td>
<td>1.12</td>
</tr>
<tr>
<td>7</td>
<td>1.04</td>
<td>1.07</td>
<td>1.09</td>
<td>1.17</td>
<td>1.24</td>
</tr>
<tr>
<td>8</td>
<td>0.98</td>
<td>1.02</td>
<td>1.1</td>
<td>1.11</td>
<td>1.2</td>
</tr>
<tr>
<td>9</td>
<td>0.97</td>
<td>1.03</td>
<td>1.12</td>
<td>1.19</td>
<td>1.26</td>
</tr>
<tr>
<td>10</td>
<td>0.98</td>
<td>1.06</td>
<td>1.13</td>
<td>1.18</td>
<td>1.28</td>
</tr>
<tr>
<td>11</td>
<td>0.95</td>
<td>1.04</td>
<td>1.03</td>
<td>1.14</td>
<td>1.19</td>
</tr>
<tr>
<td>12</td>
<td>0.98</td>
<td>1.07</td>
<td>1.07</td>
<td>1.15</td>
<td>1.15</td>
</tr>
<tr>
<td>13</td>
<td>0.99</td>
<td>0.95</td>
<td>1.11</td>
<td>1.17</td>
<td>1.1</td>
</tr>
<tr>
<td>14</td>
<td>1.01</td>
<td>1.1</td>
<td>1.05</td>
<td>1.13</td>
<td>1.18</td>
</tr>
<tr>
<td>15</td>
<td>1.04</td>
<td>1.1</td>
<td>1.05</td>
<td>1.15</td>
<td>1.11</td>
</tr>
<tr>
<td>16</td>
<td>0.99</td>
<td>0.97</td>
<td>1.08</td>
<td>1.11</td>
<td>1.22</td>
</tr>
<tr>
<td>17</td>
<td>1.04</td>
<td>1.1</td>
<td>1.11</td>
<td>1.15</td>
<td>1.24</td>
</tr>
<tr>
<td>18</td>
<td>0.93</td>
<td>1.06</td>
<td>1.07</td>
<td>1.13</td>
<td>1.14</td>
</tr>
<tr>
<td>19</td>
<td>0.96</td>
<td>0.98</td>
<td>1.12</td>
<td>1.14</td>
<td>1.21</td>
</tr>
<tr>
<td>20</td>
<td>1.08</td>
<td>1.04</td>
<td>1.09</td>
<td>1.11</td>
<td>1.13</td>
</tr>
</tbody>
</table>

Table 3

The data of worker’s level of skill for each product type \((\beta_{im})\).

<table>
<thead>
<tr>
<th>Product type</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>N(1,0.05)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N(1.05,0.05)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N(1.1,0.05)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N(1.15,0.05)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N(1.2,0.05)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4

The data of worker’s level of skill \((\beta_{im})\).

<table>
<thead>
<tr>
<th>Worker/Product</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6 7 8 9 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5

The data of 30 batches.

<table>
<thead>
<tr>
<th>Batch number</th>
<th>Product type</th>
<th>Batch size ((B_m))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>53</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>54</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>49</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>49</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>55</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>54</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>48</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>48</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>48</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>46</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>58</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>48</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>52</td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>48</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>51</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>54</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>57</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>54</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>49</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>53</td>
</tr>
<tr>
<td>22</td>
<td>3</td>
<td>46</td>
</tr>
<tr>
<td>23</td>
<td>4</td>
<td>45</td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td>46</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
<td>45</td>
</tr>
<tr>
<td>26</td>
<td>3</td>
<td>44</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>53</td>
</tr>
<tr>
<td>28</td>
<td>4</td>
<td>47</td>
</tr>
<tr>
<td>29</td>
<td>2</td>
<td>53</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>52</td>
</tr>
</tbody>
</table>
$$d(z', z) = \max_{j=1}^{r} \left( \frac{1}{\lambda_j} (z'_j - z_j) \right), \quad z' = (z'_1, \ldots, z'_r) \in ND,$$

$$z = (z_1, \ldots, z_r) \in R \quad (18)$$

where $\lambda_j$ is the range of the objective $F_j$ among all reference and heuristic solutions.

Note that $D_{av}$ is the average distance from a solution $z \in R$ to its closest solution in $ND$ while $D_{max}$ yields the maximum of the minimum distance from a solution $z \in R$ to any solution in $ND$ (Arroyo & Armentano, 2005).

We repeatedly ran the algorithm 100 times to solve the two instances with 5 and 6 workers. The performance comparative results to the Enumeration are shown in Tables 6 and 7, respectively.

Table 6 shows the result of the problem with 5 workers after executing our algorithm 100 times. The elements in Table 6 are the number of true non-dominated solutions, the accuracy rate, the average (minimum) RNI, average $D_{av}$, average $D_{max}$ and average computational time with the of different $N$ and $n$. The larger $Ac$ Rate, $Av$ RNI and $Min$ RNI, the better the non-dominated solutions produced by our algorithm; the less $Av$ $D_{av}$ and $Av$ $D_{max}$, the closer the solution of our algorithm to the true non-dominated front. For example, if $Ac$ Rate is equal to 100, at the same time $Min$ RNI, $Av$ RNI, $Av$ $D_{av}$, and $Av$ $D_{max}$ would be 1, 1, 0 and 0, respectively. This means each non-dominated front produced by the algorithm among 100 times is the same as the result from enumeration.

From Table 6, we can conclude that the increase of $N$ and/or $n$ may obtain better solutions. This result illustrates the reliability of the developed algorithm.

Table 7 shows the performance of the algorithm for the instance with 6 workers. From Table 7, we may observe that the increase of $N$ is more effective than $n$ for the performance of the algorithm. Again, the result in Table 7 illustrates the reliability of the developed algorithm.

According to Theorem 6 and Eq. (13), it can be inferred that a larger $W$ will lead more feasible solutions. From Tables 6 and 7, we can also observe that the increase of $W$ will decrease the performance of the algorithm.

### Results of larger problems

We run the algorithm to solve larger instances with 10, 15 and 20 workers and their feasible solutions are 102,247,563, 230,283, 190,977,853 and 2,677,687,796,244,384,203,115, respectively.

Because true non-dominated solutions are unknown, the performance is evaluated with respect to the reference set $R$, which consists of the non-dominated solutions generated by running the algorithm 100 times for each instance. The performances of the larger instances with 10, 15 and 20 workers relative to reference set $R$ are shown in Table 8.
and average computational times show that non-dominated front produced by the algorithm produces few efficient solutions. The values of the reference set every non-dominated front produced by the algorithm is equal to the algorithm is not close to the reference set. However, running times are less than 1, which represents that not is small and close to 0, which represents the possibility of producing better non-dominated solutions. Third, we have developed a NSGA-II based algorithm to solve the multi-objective problems. We modified several implementation operators in the original algorithm of NSGA-II (Deb et al., 2002) to fit with the features of the line-cell conversion problem.

For each problem size, Table 8 shows the number of reference true non-dominated solutions. $N$: number of changing non-dominated front, $n$: population size of the algorithm. $Ac$ rate: accuracy rate of the non-dominated front produced by the proposed algorithm among 100 times. $Av$ and $Min$: average and minimum, ss: seconds.

For the performances for the instances with 10, 15 and 20 workers, and the results are shown in Table 9.

The $Av$ RNI and $Max$ RNI in Table 9 are much larger than those in Table 8, and $Av$ $D_{av}$ and $Av$ $D_{max}$ are close to 0 and less than those in Table 8. These mean that the non-dominated front produced by the algorithm with $T(10)$ is close to the reference and a good sub-optimal non-dominated front is obtained. In addition, the computational time increases non-exponentially.

Therefore, increasing $P_c$, $P_m$, $N$, $n$ and $T$ may obtain better non-dominated solutions.

### 6. Conclusions

The line-cell conversion has been implemented in Japan, US (Williams, 1994), Europe and Korea (Yin, 2006), China (Cao, 2002) to fit with the features of the line-cell conversion problem. Other countries (Yin, Stecke et al., 2011). A seru system has many benefits. It can reduce lead time, setup time, WIP inventories, finished-product inventories, cost, required workforce, shop floor space, required energy, and CO$_2$. Seru also influences profits, product quality, and workforce motivation in a positive way. All these performance results need to be investigated from various perspectives. The inherent mechanisms that explain why seru can help improve performances need to be identified. Also, effective algorithms need to be developed to show how to efficiently improve these performances.

Unfortunately, the research in this area is relatively lacking. To the best of our knowledge, most studies, e.g., Liu et al. (2013) and Yu et al. (2012), are attempting to develop algorithms to form serus and assign workers to formed serus. Only one paper (Yin, Stecke et al., 2011) creates rigorous mathematical model to find mechanism why seru can reduce required workforce. Our current paper is the first study that tries to uncover the underlying mechanism why a seru system can improve two performances, i.e., the total throughput time (TTP) and the total labor hours (TLH). Other contributions of this study are as follows.

First, we have clarified the solution space and combinatorial complexity of the line-cell conversion problem based on the modified simple model of Kaku et al. (2008) in which an assembly line is converted to a pure cell system. Our results obtained in this research may be still true for more complex systems, i.e., the original line-cell conversion problem of Kaku et al. (2008). Second, we have proved several theorems, which show that the line-cell conversion problem is NP-hard and the non-dominated or Pareto-optimal front of the multi-objective line-cell conversion problem is non-convex. Third, we have developed a NSGA-II based algorithm to solve the multi-objective problems. We modified several implementation operators in the original algorithm of NSGA-II (Deb et al., 2002) to fit with the features of the line-cell conversion problem.
Comparing with the results obtained by Enumeration for small size examples, our NSGA-II worked well for the line-cell conversion problem. We also tested it on larger line-cell conversion problems and found that the non-dominated solutions are converged.

For the future research, there are a lot of research problems. A thorough unanswerable list can be found in Yin et al. (2012), such as partially cross-trained workers (i.e., a worker cannot perform all assembly tasks), different products have different assembly tasks, cost of karakuri (i.e., duplication of equipment), human and psychology factors, environmental considerations in different countries (Dong, Liu, & Dong, in press; Muldul & Barve, in press), and so on.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (71021061, 61203182), the National Social Science Foundation of China (13CGL045), the China Postdoctoral Science Foundation (2013T60296, 2012M510828), and the project of Sanqin Scholars of Shaanxi Province.

References


